04_homework_linear_regression

November 18, 2017

1 Programming assignment 4: Linear regression

```
In [1]: import numpy as np

        from sklearn.datasets import load_boston
        from sklearn.model_selection import train_test_split
```

1.1 Your task

In this notebook code skeleton for performing linear regression is given. Your task is to complete the functions where required. You are only allowed to use built-in Python functions, as well as any `numpy` functions. No other libraries / imports are allowed.

1.2 Load and preprocess the data

I this assignment we will work with the Boston Housing Dataset. The data consists of 506 samples. Each sample represents a district in the city of Boston and has 13 features, such as crime rate or taxation level. The regression target is the median house price in the given district (in $1000's).

More details can be found here: http://lib.stat.cmu.edu/datasets/boston

```
In [2]: X , y = load_boston(return_X_y=True)

        # Add a vector of ones to the data matrix to absorb the bias term
        # (Recall slide #7 from the lecture)
        X = np.hstack([np.ones([X.shape[0], 1]), X])
        # From now on, D refers to the number of features in the AUGMENTED dataset (i.e. inclu

        # Split into train and test
        test_size = 0.2
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
```

1.3 Task 1: Fit standard linear regression

```
In [8]: def fit_least_squares(X, y):
            """Fit ordinary least squares model to the data.

            Parameters
```

```
    ----------
    X : array, shape [N, D]
        (Augmented) feature matrix.
    y : array, shape [N]
        Regression targets.

    Returns
    -------
    w : array, shape [D]
        Optimal regression coefficients (w[0] is the bias term).

    """
    w_best = np.dot(np.dot(np.linalg.inv(np.dot(X.T, X)), X.T), y)
    return w_best
```

1.4 Task 2: Fit ridge regression

```
In [10]: def fit_ridge(X, y, reg_strength):
             """Fit ridge regression model to the data.

             Parameters
             ----------
             X : array, shape [N, D]
                 (Augmented) feature matrix.
             y : array, shape [N]
                 Regression targets.
             reg_strength : float
                 L2 regularization strength (denoted by lambda in the lecture)

             Returns
             -------
             w : array, shape [D]
                 Optimal regression coefficients (w[0] is the bias term).

             """
             N, D = X.shape
             w_best = np.dot(np.dot(np.linalg.inv(np.dot(X.T, X) + reg_strength * np.eye(D)), X
             return w_best
```

1.5 Task 3: Generate predictions for new data

```
In [11]: def predict_linear_model(X, w):
             """Generate predictions for the given samples.

             Parameters
             ----------
             X : array, shape [N, D]
                 (Augmented) feature matrix.
```

```
        w : array, shape [D]
            Regression coefficients.

        Returns
        -------
        y_pred : array, shape [N]
            Predicted regression targets for the input data.

        """
        preds = np.dot(X, w)
        return preds
```

1.6 Task 4: Mean squared error

```
In [12]: def mean_squared_error(y_true, y_pred):
             """Compute mean squared error between true and predicted regression targets.

             Reference: `https://en.wikipedia.org/wiki/Mean_squared_error`

             Parameters
             ----------
             y_true : array
                 True regression targets.
             y_pred : array
                 Predicted regression targets.

             Returns
             -------
             mse : float
                 Mean squared error.

             """
             N = y_true.shape[0]
             diff = y_pred - y_true
             MSE = (1.0/N) * np.dot(diff.T, diff)
             return MSE
```

1.7 Compare the two models

The reference implementation produces * MSE for Least squares $\approx$ **23.98** * MSE for Ridge regression $\approx$ **21.05**

You results might be slightly (i.e. $\pm 1\%$) different from the reference soultion due to numerical reasons.

```
In [13]: # Load the data
         np.random.seed(1234)
         X , y = load_boston(return_X_y=True)
         X = np.hstack([np.ones([X.shape[0], 1]), X])
```

```
        test_size = 0.2
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)

        # Ordinary least squares regression
        w_ls = fit_least_squares(X_train, y_train)
        y_pred_ls = predict_linear_model(X_test, w_ls)
        mse_ls = mean_squared_error(y_test, y_pred_ls)
        print('MSE for Least squares = {0}'.format(mse_ls))

        # Ridge regression
        reg_strength = 1
        w_ridge = fit_ridge(X_train, y_train, reg_strength)
        y_pred_ridge = predict_linear_model(X_test, w_ridge)
        mse_ridge = mean_squared_error(y_test, y_pred_ridge)
        print('MSE for Ridge regression = {0}'.format(mse_ridge))
```

```
MSE for Least squares = 23.9843076118
MSE for Ridge regression = 21.0514870338
```

Problem 1

See above.

Problem 2

$$\begin{aligned} E_{weighted}(w) &= \frac{1}{2}\sum_{i=1}^{N} t_i \cdot [w^T\phi(x_i) - y_i]^2 \\ &= \frac{1}{2}(\Phi w - y)^T T(\Phi w - y) \qquad \text{(T = diag}(t_i)\text{, i = [1,...,N])} \end{aligned}$$

$$\begin{aligned} \frac{\partial E_{weighted}(w)}{\partial w} &= \frac{\partial \frac{1}{2}[(\Phi w)^T T\Phi w - (\Phi w)^T Ty - y^T T\Phi w + y^T Ty]}{\partial w} = 0 \\ &\Leftrightarrow \frac{\partial E_{weighted}[\frac{1}{2}(w^T\Phi^T T\Phi w - 2y^T T\Phi w + y^T Ty)]}{\partial w} = 0 \\ &\Leftrightarrow \Phi^T T\Phi w - \Phi^T Ty = 0 \qquad \text{(by } \tfrac{\partial x^T a}{\partial x} = \tfrac{\partial a^T x}{\partial x} = a \text{ and } \tfrac{\partial x^T Bx}{\partial x} = (B + B^T)x\text{)} \\ w^* &= (\Phi^T T\Phi)^{-1}\Phi^T Ty \end{aligned}$$

The weighting factors t_i can (i) reduce the variance of the noise on the data by "pulling back" the predictions which are far from their actual label and (ii) artificially create different datapoints from exact copies of the data by using different values for the respective t_i or remove them by setting the respective t_i to zero.

Problem 3

Let $\hat{\Phi} \in \mathbb{R}^{(N+M)\times M}$ and $\tilde{\Phi} \in \mathbb{R}^{(N+M)\times M}$ where the last M are $\mathbf{0}I_M$ resp. the first N rows are $\mathbf{0}I_N$.

$$\begin{aligned} E_{LS}(w) &= \frac{1}{2}(\Phi w - y)^T(\Phi w - y) \\ &= \frac{1}{2}[(\Phi w)^T\Phi w - 2y^T\Phi w + y^T y] \\ &= \frac{1}{2}[(\tilde{\Phi} w)^T\tilde{\Phi} w + (\hat{\Phi} w)^T\hat{\Phi} w + y^T y] \\ &= \frac{1}{2}[(\tilde{\Phi} w)^T\tilde{\Phi} w + y^T y + \lambda||w||^2] \\ &= E_{rigde}(w) \end{aligned}$$

Thus the ridge regression estimates can be obtained by deriving the least-squares estimates on the augmented dataset.

Problem 4

$$
\begin{aligned}
p(w,\beta|\mathcal{D}) &\propto (\beta^{\frac{N}{2}} e^{\frac{-\beta}{2}\sum_{i=1}^{N}(y_i - w^T\phi(x_i))^2}) \cdot (e^{\frac{-\beta}{2}[(w-m_0)^T S_0^{-1}(w-m_0)+2b_0]}\beta^{\frac{M}{2}+a_0-1}) \\
&= \beta^{\frac{M+N}{2}+a_0-1} e^{[\frac{-\beta}{2}\sum_{i=1}^{N}(y_i - w^T\phi(x_i))^2 + (w-m_0)^T S_0^{-1}(w-m_0)+2b_0]} \\
&= \beta^{\frac{M+N}{2}+a_0-1} \underbrace{e^{\frac{-\beta}{2}(y-\Phi w)^T(y-\Phi w)+(w-m_0)^T S_0^{-1}(w-m_0)+2b_0]}}_{(1)}
\end{aligned}
$$

$$
\begin{aligned}
&= (y-\Phi w)^T(y-\Phi w) + (w-m_0)^T S_0^{-1}(w-m_0) + 2b_0 \\
&= -2y^T\Phi w - 2m_0^T S_0^{-1} w + w^T\Phi^T\Phi w + w^T S_0^{-1} w + y^T y + m_0^T S_0^{-1} m_0 + 2b_0 \\
&= -2(y^T\Phi + m_0^T S_0^{-1})w + w^T(\Phi^T\Phi + S_0^{-1})w + y^T y + m_0^T S_0^{-1} m_0 + 2b_0 \\
&= -2(y^T\Phi + m_0^T S_0^{-1})S_N S_N^{-1} w + w^T S_N^{-1} w + y^T y + m_0^T S_0^{-1} m_0 + 2b_0 \qquad (S_N = (\Phi^T\Phi + S_0^{-1})^{-1}) \\
&= -2\mu_N^T S_N^{-1} w + \mu_N^T S_N^{-1}\mu_N - \mu_N^T S_N^{-1}\mu_N + w^T S_N^{-1} w + y^T y + m_0^T S_0^{-1} m_0 + 2b_0 \\
&\qquad\qquad (\mu_N = S_N(S_0^{-1}\mu_0 + \Phi^T y)) \\
&= \underbrace{(w-\mu_N)^T S_N^{-1}(w-\mu_N) - \mu^T S_N^{-1}\mu + y^T y + m_0^T S_0^{-1} m_0 + 2b_0}_{(2)}
\end{aligned}
$$

$$
\begin{aligned}
p(w,\beta|\mathcal{D}) &\propto \beta^{\frac{M+N}{2}+a_0-1} \cdot e^{\frac{-\beta}{2}(w-\mu_N)^T S_N^{-1}(w-\mu_N) - \mu^T S_N^{-1}\mu + y^T y + m_0^T S_0^{-1} m_0 + 2b_0} \qquad \text{(by plugging (2) in (1))} \\
&= \beta^{\frac{M}{2}+a_n-1} \cdot e^{\frac{-\beta}{2}(w-\mu_N)^T S_N^{-1}(w-\mu_N)+2b_n} \\
&\qquad\qquad \text{(by } a_n = \tfrac{N}{2} + a_0,\ b_n = \tfrac{1}{2}(-\mu^T S_N^{-1}\mu + y^T y + m_0^T S_0^{-1} m_0) + b_0) \\
&\propto \mathcal{N}(w|\mu_n, \beta^{-1} S_n) \cdot \text{Gamma}(\beta|a_n, b_n)
\end{aligned}
$$